Weigh the benefits of fuzzy-logic vs classical control in a disk-drive spindle

Brian P Tremaine, Seagate Technology

You can apply fuzzy- and classical-control techniques to any servo-control loop. Which technique you use depends heavily on the nonlinearities in a system.

A furor is raging over whether you should use fuzzy-logic or classical-control tools to analyze a complex servo-control system. Fuzzy-logic advocates claim it eliminates the need for a mathematical system description, while classical-control advocates claim fuzzy logic's lack of analysis tools makes it undesirable. To observe these extreme views, compare the classical- and fuzzy-control analysis of a disk-drive dc spindle motor. The comparison uses a C program, which simulates the plant and classical feedback-control signal.

The FIDE (fuzzy inference-development) program from Aptronix Inc compiles the plant program to simulate a fuzzy controller. The spindle-motor example shows that even a basic system can have nonlinearities, which preclude linear analysis. However, knowledge of the system equations can aid the design of a fuzzy controller.

The dc spindle motor, driver electronics, and digital transducers comprise the system's plant and are inherently nonlinear. The driver develops only positive torque, which is a common practice in disk-drive applications. Therefore, the motor depends only on coulomb friction to decelerate. The plant's nonlinearities include saturation of the error signal, torque control in only one quadrant, and quantization of the transducer signals from the output control voltage. Therefore, linear-analysis methods do not apply for either the classical or fuzzy set of control tools. This analysis compares the differences between a classical proportional plus integral (PI) controller and a fuzzy controller.

The plant model (**Fig 1**) is a dc 3-phase, 12-pole spindle motor along with the driver electronics. The output circuitry comprises three half-bridge drivers with current-sense feedback. The input to the driver is an analog voltage, which generates the spindle motor current through a current amplifier having a gain of gm (amps/volt). When the input voltage equals some reference voltage, the current command is zero, and the output stages are off. During this time, coulomb friction decelerates the motor. Maximum current drive occurs when the input voltage is at 0V. The motor resistance, supply voltage, and the back EMF from the motor limit the maximum current drive to the motor.

Simulator doesn't model current control

The current-control loop around the motor and the current amplifier typically have a bandwidth several orders of magnitude greater than the motor's velocity bandwidth. There-

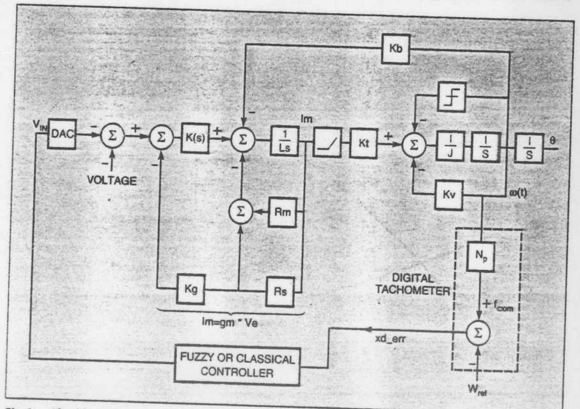

Fig 1—The block diagram of the plant for a 3-phase, 12-pole spindle motor comprises a current amplifier driving the motor and half-bridge output drivers with current-sensing feedback.

FUZZY AND CLASSICAL CONTROL

fore, the plant simulation doesn't include the dynamics of the current-control loop in **Fig 1**. Instead, the plant model calculates the maximum available current and drives the motor using a current source, which is equal to either the commanded current or the maximum available current, whichever is smaller.

The classical-control analysis applies linearization to the nonlinear plant to design a proportional PI controller. The linear model does not predict the correct simulated transient response. The simulated plant model calculates motor speed in rad/sec, position in rad, and a digital encoder pulse, called f_{com}. A digital tachometer and digital phase detector use the f_{com} signal, which occurs nine times/revolution of the motor (N_p=9). The linearized transfer function of the motor and current driver in continuous time is:

$$H(s)=(gm\cdot Kt/Jm)/(s+Kv/Jm),\ rad/sec\cdot V, \qquad (1)$$

where gm=0.5 A/V, transconductance; Kt=1.05 oz-in./A, torque constant; Kv=0.001 oz-in./sec, viscous constant; and Jm=752E–6 oz-in.-sec^2, inertia.

The equation represents the transfer function from the input voltage to the motor's radial velocity output. Because the tachometer's outputs are discrete, the continuous-time representation is converted to discrete time using the z-transform. The transform involves an approximation because the sample interval is at a fixed motor rotation ($2\pi/N_p$) and not at a fixed sample time. As a practical matter, the discrete system is often analyzed assuming T is a fixed sample time. Although we make this approximation, linearizing the system and assuming a fixed sample time completely ignores the real system's nonlinearity. Stability of the linearized system does not ensure global stability of the real system. An appropriate Lyaponov function could determine a nonlinear system's degree of stability. Finding such a Lyaponov function is difficult, however.

You can derive a difference equation directly from **Eq 1** by assuming a small time increment, as follows:

$$(\omega(t+dt)=(1-dt\cdot(Kv/Jm))\cdot\omega(t)+(gm\cdot Kt\cdot dt/Jm)\cdot v(t). \qquad (2)$$

If dt=T and t=kT, then **Eq 2** becomes

$$(\omega(kT+T=(1-T\cdot(Kv/Jm))\cdot\omega(kT)+(gm\cdot Kt\cdot T/Jm)\cdot v(kT).$$

Noting that the z-transform of ω(kT+T is ω(z)·z, the overall transfer function is

$$H(z)=gm\cdot Kt\cdot T/Jm)/(z-(1-T\cdot Kv/Jm)),\ rad/sec\cdot V. \qquad (4)$$

Digital tachometer provides feedback

Regardless of whether the controller is classical or fuzzy, you must measure the spindle speed for feedback control. **Fig 1** includes a block diagram labeled "digital tachometer" showing the measurement of the spindle speed, ω(t). The output of this block is xd_err, a digital word representing speed error.

The digital tachometer counts the period of the f_{com} pulse with 1-μsec resolution (t_i). You subtract the final count from a reference count of 1880. The tachometer then truncates the count to fall within –128 to +127 counts to generate the xd_err signal. If the time interval is T_t and the average velocity over an f_{com} period is w_{avg}, then $w_{avg}(T_t)$ is the average velocity over the time interval:

$$\omega_{avg}(T_t)=2\cdot\pi/(N_p\cdot T_t). \qquad (5)$$

This equation is nonlinear in the variable T_t. To linearize the equation, you can write a Taylor series around the reference velocity ω_{ref} and the reference period T_t. You linearize a function using a Taylor series about an equilibrium point. If you linearize f(x,t) about the point x0, then the first two terms of the Taylor series are

$$f(x,t)=f(x0,t)+f'(x0,t)(x-x0),$$

where f' denotes the derivative with respect to x. Applying this to **Eq 5** yields

$$\omega_{avg}(T_t)=\omega_{ref}-(\omega_{ref}/T_{ref})*(T_t-T_{ref}). \qquad (7)$$

Because $\omega_{avg}(T_t)$ is the average velocity over the period T_t, you can express this equation in terms of the velocity at the beginning and the end of the period, as follows:

$$\omega_{avg}(k)=(1/2)\cdot(\omega(k)+\omega(k-1)). \qquad (8)$$

Combining **Eqs 7** and **8** yields

$$(\omega(k)-\omega_{ref}+(\omega(k-1)-\omega_{ref})-2'(\omega_{ref}/T_{ref})\cdot t_t-T_{ref}). \qquad (9)$$

Denoting the difference between w and ω_{ref} as ω_e and measuring the time interval in terms of t_i reduces **Eq 9** to

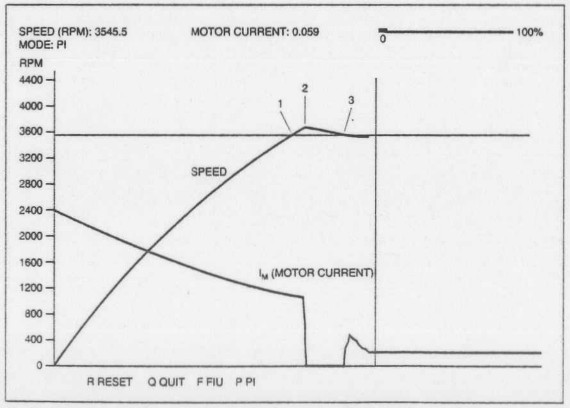

Fig 2—Driving a spindle motor from a dead stop to a final spindle speed using a classical proportional integral (PI) controller shows a severe target overshoot. The integrator shuts off the motor current at Point 2 to allow the motor to settle using coulomb friction.

FUZZY AND CLASSICAL CONTROL

$$\omega e(k)+\omega e(k-1=-2\cdot(\omega_{ref}/T_{ref})\cdot t_i\cdot d(k), \quad (10)$$

where d(k) is the digital count of the time interval.

The z-transform for **Eq 10** is the transfer function of the digital tachometer, as follows:

$$\omega e(z)+\omega e(z)/z=-2\cdot(\omega_{ref}/T_{ref})\cdot(T_t-t_i\cdot d(z).$$
$$D(z)=d(z)/\omega e(z)=-T_{ref}/(2\cdot\omega_{ref}\cdot t_i))\cdot 1+z)/z.$$

If the motor velocity is significantly different from the reference frequency (ω_{ref}), the Taylor series expansion is no longer accurate. In addition, if the velocity is low, the sample time is long, and the phase delay of D(z) is long.

The plant-transfer function in **Eq 1** has a maximum phase delay of 90°. Even when you represent **Eq 1** in discrete form (**Eq 4**), the closed-loop bandwidth is sufficiently below the Nyquist frequency, 1/T, that the phase delay caused by sampling is small. Therefore, you can stabilize the plant using only proportional feedback from the tachometer. However, the speed error is not exactly zero. Offsets in the driver stage and motor running torque relative to the driver input requires a speed error to counteract the offsets.

To force the speed error to zero, classical feedback-control systems sum the speed error and the time integral of the speed error (PI) to generate the feedback-control law. The control-law calculation for a discrete time system follows:

$$\text{Vint}(k)=\text{Vint}(k-1+k_i\cdot \text{xd\_err}(k); \quad (13)$$

$$\text{Vout}(k)=k_p\cdot \text{xd\_err}(k)+\text{Vint}(k), \quad (14)$$

where xd_err is the digital word representing speed error. **Eq 13** is the difference equation for the digital integrator, and **Eq 14** is the linear sum of the speed error and time integral of the speed error. You can combine **Eqs 13** and **14** into a transfer function in the z-domain for the equivalent digital controller, C(z), as follows:

$$C(z)=\text{Vout}(z)/\text{xd\_err}(z)=kp+ki\cdot z/(z-1).$$

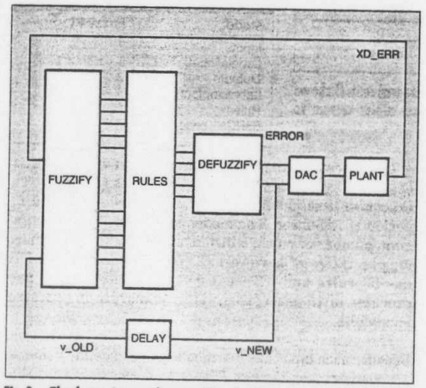

Fig 3—The fuzzy integral-control block diagram uses a fuzzy-inference unit, which is analogous to the proportional plus integral control of classical methods.

The open-loop gain of the linearized system is

$$H(z)\cdot C(z)\cdot D(z),\ Gol(z)=D(z)\cdot((k_p+k_i)\cdot gm\ K_t\cdot T/J_m)\cdot (z-k_p)/(k_p+k_i))/[z-(1-T\cdot Kv/Jm))\cdot(z-1)]. \quad (17)$$

Substituting some of the system parameters into **Eq 17** yields open-loop poles as z1=1, and z2=0.99. The digital tachometer produces a pole at z=0 and a zero at z=−1. The ratio of the integral gain to proportional gain determines the zero at $z=1/(1+k_i/k_p)$.

The linearized model predicts closed-loop poles that are well-damped for small k_i/k_p and under-damped for large k_i/k_p. This result is in contrast to the simulation results (**Fig 2**) and highlights the fact that nonlinearities in a system can invalidate the use of linear methods.

Fig 2 shows a plot of the simulated spindle speed using a

```
$ SPINDLE disk drive spindle PI control
$ brian tremaine 6 July 1993, revised 1 Jan. 1994
fiu tvfi (min max) ;

>xd_err "bits":-128()127 [
    Neg_Large  (@-128,255, @-64,0),
    Neg_Med    (@-128,0,   @-64,255, @0,0),
    Zero       (@-64,0,    @0,255,   @64,0),
    Pos_Med    (@0,0,      @64,255,  @127,0),
    Pos_Large  (@64,0,     @127,255)];

>v_old "bits":0()255 [
    Neg_Large (@0,255,   @64,0),
    Neg_Med   (@0,0,     @64,255,  @128,0),
    Zero      (@64,0,    @128,255, @192,0),
    Pos_Med   (@128,0,   @192,255, @255,0),
    Pos_Large (@192,0,   @255,255) ];

<error "bits": 0()255* (
      Neg_Large =  0.0,
      Neg_Med =      8,
      Zero =       128,
      Pos_Small =  248,
      Pos_Large =  255 );

<v_new "bits": 0()255* (
      Neg_Large =  0.0,
      Neg_Med =     64,
      Zero =       128,
      Pos_Med =    192,
      Pos_Large =  255  );
$
if xd_err is Neg_Large                          then v_new is Neg_Large;
if xd_err is Neg_Med                            then v_new is Neg_Large;
if xd_err is Zero and v_old is Pos_Large       then v_new is Pos_Large;
if xd_err is Zero and v_old is Pos_Med         then v_new is Pos_Med;
if xd_err is Zero and v_old is Zero            then v_new is Zero;
if xd_err is Zero and v_old is Neg_Med         then v_new is Neg_Med;
if xd_err is Zero and v_old is Neg_Large       then v_new is Neg_Large;
if xd_err is Pos_Med and v_old is Pos_Med      then v_new is Pos_Large;
if xd_err is Pos_Med and v_old is Zero         then v_new is Pos_Med;
if xd_err is Pos_Med and v_old is Neg_Med      then v_new is Zero;
if xd_err is Pos_Med and v_old is Neg_Large    then v_new is Neg_Med;
if xd_err is Pos_Large                          then v_new is Pos_Large;
$
if xd_err is Neg_Large                          then error is Neg_Large;
if xd_err is Neg_Med                            then error is Neg_Med;
if xd_err is Zero                               then error is Zero;
if xd_err is Pos_Med                            then error is Pos_Small;
if xd_err is Pos_Large                          then error is Pos_Large;
end
```

Fig 4—The listing for the fuzzy controller of the spindle disk drive is compiled with Aptronix's FIDE (fuzzy inference development) program.

FUZZY AND CLASSICAL CONTROL

		V_OLD				
		NEG_LARGE	NEG_MED	ZERO	POS_MED	POS_LARGE
	NEG_LARGE	NL/NL	NL/NL	NL/NL	NL/NL	NL/NL
	NEG_MED	NL/NM	NL/NM	NL/NM	NL/NM	NL/NM
XD_ERR	ZERO	NL/ZR	NM/ZR	ZR/ZR	PM/ZR	PL/ZR
	POS_MED	NM/PM	ZR/PM	PM/PM	PL/PM	/PM
	POS_LARGE	PL/PL	PL/PL	PL/PL	PL/PL	PL/PL

OUTPUT LEGEND: v_new/error
NL = NEG_LARGE
NM = NEG_MED
ZR = ZERO
PM = POS_MED
PL = POS_LARGE

Fig 5—The rule table for the fuzzy integral controller contains redundant rows of rules that can be condensed into fewer rules for the fuzzy-inference unit to follow.

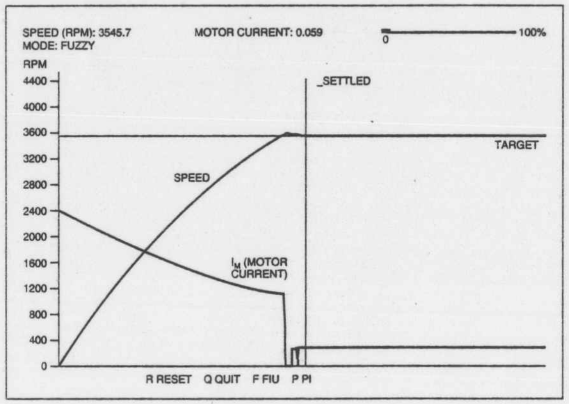

Fig 6—The spin-up of the spindle motor from a dead stop using fuzzy integral control shows less overshoot than with a classical PI controller. The integrator reduces motor current to zero when the system is over speed to cause deceleration.

classical PI controller with a nonlinear plant. In **Eqs 13** and **14**, k_i=3/2048, and k_p=48/2048. The speed overshoots the target and slowly decays with a linear ramp to the target speed value. From time t=0 until Point 1, the speed error is negative, and the integrator continues to accumulate until it saturates. Not until the speed error begins to go positive does the integrator begin to come out of saturation. At Point 2 the integrator has recovered enough to allow the control voltage to command zero current. From Point 2 until Point 3 the only deceleration is due to coulomb friction. Beyond Point 3, the system settles to a steady state value.

Fuzzy control uses different control blocks

Fuzzy integral control analyzes the spindle motor using a fuzzy-inference unit (FIU) (**Fig 3**). This analysis block is analogous to the proportional plus integral control block of the classical method. Because an FIU has no memory, fuzzy control accomplishes the integral function by using the feedback from the FIU's output to its input. The FIU has the FIDE listing shown in **Fig 4**. The inputs to the FIU are the speed error (xd_err) and previous output integral count (V_old). The outputs of the FIU are proportional count (error) and integral count (V_new). **Fig 5 shows** a table representation of the FIU rules. The control law is a summation of both outputs

$$Vout=16\cdot(error-128)/255+3 \cdot (v\_new-128)/255+3.$$

Fig 6 shows a plot of motor speed and current for spin-up from a dead stop. The steady state speed nearly equals the commanded speed, as in the integral controller. However, the fuzzy method improves transient response, decreases overshoot, and provides faster settling. A classical PI system involves a tradeoff between overshoot and settling time. The fuzzy-system rules and membership functions effectively implement "anti-windup" integral control. When xd_err is positive, (over speed) the integral term (v_new) is forced positive to zero the spindle-motor current immediately to cause deceleration.

Table 1 details the fuzzy model:

Table 1—The fuzzy model

Model	Fuzzy PI
Inputs	2
Labels/input	5
Outputs	2
Labels/output	5
Rules	17
68HC11 code bytes	717

The execution time for the above model with a 68HC11 µP is about 1.25 msec using a 50-nsec bus cycle time. The execution time is sufficient for the sample time and f_{com} period of 1.88 msec. The model is optimized for the minimum number of rules, which is an important step affecting the choice of hardware. The initial pass of the FIU has 30 rules and 877 code bytes. Although this pass executes in 1.6 msec, there is little margin in processor bandwidth.

The output v_new is a function of xd_err and v_old. Because each input has five labels, you can define v_new as a 5×5 matrix, or 25 rules. The output, error, is a function of xd_err only. Because xd_err has five labels, five rules can define the output, error. Therefore, there are 30 rules for the the first pass. In the last row of **Fig 5**, when xd_err

FUZZY AND CLASSICAL CONTROL

is Pos_Large, V_new is Pos_Large, regardless of v_old. This condition lets you replace five rules with one rule: "If xd_err is Pos_Large, then v_new is Pos_Large." Modifications such as these can reduce the number of rules from 30 to 17.

In comparing a classical PI servo-control loop with a fuzzy-control loop for a dc spindle motor, the plant and the tachometer have inherent nonlinearities that preclude applying linear stability theory, regardless of the control method. Knowledge of classical PI servo-control principles helps with the design of a fuzzy controller. The performance in terms of the spin-up settling time and the transient response is superior using the fuzzy system compared with the classical PI controller when there are severe nonlinearities.

The C source code for both the FIDE fuzzy modules and the plant model is available from the author. In addition to the PI controller, you can also use the Aptronix bulletin-board system to access source and executable simulation code for proportional control, PLL control, and dual-phase and tachometer loops for classical and fuzzy control. EDN

References

1. Kosko, B, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall, 1992.

2. Franklin, G, D Powel, and M Workman, *Digital Control of Dynamic Systems*, Second Edition, Addison Wesley, 1990.

Author's biography

Brian P Tremaine is a senior engineering director for Seagate Technology in Scotts Valley, CA. His job includes managing a group that is responsible for control-system design of actuator and spindle servos on 2.5- and 1.8-in. disk files. In his nine years with Seagate, he has helped to develop a host of disk-file products. Tremaine has BSEE and MSEE degrees from San Jose State University, San Jose, and an MBA from Golden Gate University, San Francisco. He is a registered professional engineer in California, and he is completing an engineer degree at Stanford University, Stanford, CA, this year. Tremaine is married and has two children.

VOTE

Please use the Information Retrieval Service card to rate this article (circle one):

High Interest	Medium Interest	Low Interest
586	587	588